Jens Tobias Kaarud, Magnus Falkenberg Nordvik,
and Håkon Rosseland Paulsen

Drone-based Detection of Sheep using Thermal and Visual Cameras: A Complete Approach

Master's thesis in Informatics

Supervisor: Professor Svein-Olaf Hvasshovd

June 2020

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Kunnskap for en bedre verden

Jens Tobias Kaarud, Magnus Falkenberg Nordvik, and
Håkon Rosseland Paulsen

Drone-based Detection of Sheep using Thermal and Visual Cameras: A Complete Approach

Master's thesis in Informatics
Supervisor: Professor Svein-Olaf Hvasshovd
June 2020

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU

Kunnskap for en bedre verden

Summary

There has been a considerable increase in the use of drone technology worldwide [1]. The use of unmanned aerial vehicles in many different types of professional applications has become increasingly popular over the years, including in farming-related areas [2].

In late fall, a sheep farmer has to collect all their sheep from grazing and bring them into their barn. Most of the sheep come when called, but a small number will often remain in the grazing area. Finding the last few remaining sheep can be a challenging task for a sheep farmer. The sheep are usually spread over a vast area, which is often very difficult to traverse and search. If the farmer could employ an automated system for finding these last few sheep, they could save a huge amount of time and effort.

Previous work at NTNU has primarily concerned itself with the application of object detection to find sheep in visual images [3]. This work formed the basis of a part of this thesis, as the findings were used as a starting point for the visual detection system. This thesis sets out to improve upon the previous work, combine visual detection with detection based on thermal images, and to create a complete system based on these technologies.

This thesis examines how previous work can be combined into an effective working system for finding the last sheep. It thoroughly compares different configurations of detection models and hardware solutions and evaluates how they could be used in a real-life solution.

It was found that YOLOv3-tiny is an object detection system which detects sheep very accurately, and it is also much faster than other tested configurations. This was determined to be an optimal configuration for object detection, considering hardware limitations that could be encountered in real-world usage of such a system. This method is fast enough to allow for practical use of the system ”in the field” on a laptop computer.

By splitting the input images into smaller parts, detection accuracy was greatly improved at high altitudes, and visual detection works well up to the legal maximum altitude for drone flights. Thermal detection was limited to a lower altitude by the resolution of the thermal sensor which was available. A higher-resolution thermal image would significantly increase the flight altitude at which detection could be successful, and therefore also increase the area covered by each image.

By using the metric of detecting at least one sheep in every group, the system can locate every group in the test data set; however, the real-world accuracy remains unknown. This metric was found to allow for a higher detection threshold, which reduces the number of false positives. However, entire groups of sheep could be missed if the threshold is too high, so the real-world usefulness of this method remains uncertain.

It was found that the combination of thermal and visual images can enhance the accuracy of detection, as one method often works better in situations where the other has weaknesses. Technical limitations meant that a true composition of the image types was not possible, but a simple combination of detection data from both image types still showed a significant accuracy increase.

Sammendrag

Det har vært en stor økning i bruken av dronebasert teknologi på verdensbasis [1]. Bruken av disse verktøyene i mange profesjonelle områder har blitt mer og mer populær, inkludert innen jordbruket [2].

Sent på høsten må sauebønder samle inn alle dyrene fra beitet og inn på gården. Flesteparten av sauene kommer av seg selv, men en viss andel vil ofte forbli i beiteområdet. Arbeidet med å finne de siste sauene er ofte en vanskelig oppgave for sauebonden, ettersom de kan være spredt over et veldig stort område som er vanskelig å bevege seg gjennom og lete i. Hvis bonden kunne tatt i bruk et automatisk system for å finne de siste sauene kunne de spart veldig mye tid og arbeid.

Tidligere arbeid ved NTNU har hovedsakelig omfattet bruken av objektdeteksjon for å finne sau i visuelle bilder [3]. Dette arbeidet var grunnlaget for én del av denne oppgaven, da funnene ble brukt som et startpunkt for det visuelle deteksjonssystemet. Hensikten med denne masteroppgaven er å jobbe videre med dette tidligere arbeidet, kombinere visuell deteksjon med deteksjon basert på termiske bilder, og å lage et komplett system basert på disse teknologiene.

Denne masteroppgaven undersøker hvordan tidligere arbeid kan kombineres for å lage et effektivt, fungerende system for å finne de siste sauene. Oppgaven sammenligner grundig forskjellige konfigurasjoner av deteksjonsmodeller og maskinvareløsninger, og evaluerer hvordan de kan brukes i virkelige problemstillinger.

Det ble konkludert at YOLOv3-tiny er et objektdeteksjonssystem som er veldig treffsikkert i deteksjon av sauer, og som er mye raskere enn andre testede konfigurasjoner. Det ble vist at dette er en optimal konfigurasjon for objektdeteksjon, med tanke på maskinvarebegrensninger som kan være problematiske for praktisk bruk av et slikt system. Denne metoden er rask nok for å kunne praktisk brukes lokalt på en bærbar PC.

Ved å dele bildedataen i mindre deler ble deteksjonstreffsikkerhet forbedret ved flyvning høyt over bakken, og visuell deteksjon fungerer bra opp til den lovlige maksimumshøyden for droneflyvninger. Termisk deteksjon var begrenset til en lavere høyde på grunn av oppløsningen på det termiske kameraet som var tilgjengelig. Et termisk kamera med

høyere oppløsning ville ført til en stor økning i høyden der deteksjon kan gjennomføres, og derfor også gjøre at hvert bilde dekker et større område.

Ved å bruke kriteriet at man detekterer minst én sau i hver gruppe kan systemet finne hver gruppe i testdatasettet. Det er usikkert hvor bra det fungerer i virkelig bruk. Dette kriteriet tillater en høyere deteksjonsterskel, som reduserer antallet falske positiver. Hvis denne terskelen er for høy vil hele grupper med sauer ikke bli funnet, så det gjenstår å se hvor nyttig denne metoden faktisk er.

Det ble funnet at kombinasjonen av termiske og visuelle bilder kan øke deteksjonstreffsikkerheten, da den ene metoden ofte fungerer bedre der den andre har svakheter. Tekniske begrensninger betyr at en sann kombinasjon av bildetypene ikke var mulig, men en simpel kombinasjon av deteksjonsdata fra begge viste uansett en stor økning i treffsikkerhet.

Preface

This thesis was worked on and written as our master's thesis in Informatics at the Norwegian University of Science and Technology during the year 2019 - 2020.

We want to thank Svein-Olaf Hvasshovd for his help and guidance during the process. Jonas Hermansen Muribø's previous master thesis has been beneficial for our research, and Kari Meling Johannessen provided us with the code to correlate coordinate locations for the different cameras.

We would also like to show our gratitude to NTNU HPC Group that has given us access to their powerful equipment to train our detection models. The labeling of our data was done manually using an online collaboration tool called LabelBox [4], which made the labeling process a better experience.

Table of Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
CNN	Convolutional Neural Network
COCO	Common Objects in Context
CPU	Central Processing Unit
DPM	Deformable Part Model
FLOP	Floating-Point Operation
FOV	Field Of View
FPS	Frames Per Second
GPS	Global Positioning System
GPU	Graphics Processing Unit
HPC	High Performance Computing
HTTP	HyperText Transfer Protocol
ILSVRC	ImageNet Large Scale Visual Recognition Competition
IoU	Intersection Over Union
IP	Internet Protocol
mAP	Mean Average Precision
R-CNN	Regions with CNN Features
REST	Representational State Transfer
SDK	Software Developer Kit
SVM	Support Vector Machine
UAV	Unmanned Aerial Vehicle
UI	User Interface
UWP	Universal Windows Platform

VOC	Visual Object Classes
YOLO	You Only Look Once

Chapter 1

Introduction

Sheep husbandry in Norway is largely based on the use of open ranges. The sheep and lambs are released in spring and are free to roam unherded until autumn, when some lambs are slaughtered and the rest are brought in to breed and give birth to next year's lambs [5].

Norwegian sheep farmers are legally required to round up all their animals at the end of every season [6]. While most of the sheep stay together in large groups and can be easily found, there is usually a small number of sheep which have become separated from the group. These sheep can require a lot of work to find. Modern technology, including drones, is sometimes used to aid in this effort, but manually locating the sheep is still a considerable effort for the farmer.

Sheep are highly social animals, who usually graze together and move as a group. This instinct is strong in wild sheep, but has been even further enhanced in domesticated animals [7]. Consequently, detecting a single sheep in a group can often be sufficient, as the rest will usually then be close enough for manual detection by the farmer. Detection of at least one sheep in the group can therefore be an alternative metric for the success of a sheep detection system.

In recent years drone technology has seen an explosion in usage worldwide, as consumer-oriented systems have become cheaper and much more capable [8]. These systems can be used for applications that would traditionally require a helicopter or other manned aircraft, and have made such applications much more viable for the average person. A simple camera-equipped drone can now be purchased at an affordable price, and can be used with minimal license requirements [9].

The rise of artificial intelligence and machine learning technology has been at the core of huge leaps in the capabilities of image recognition and object detection technologies in recent years [10]. Some object detection systems allow a computer to search through images for objects at a much faster rate than can be done manually by a human, and

accuracy rates in some data sets are now approaching or surpassing human levels [11].

The speed at which object detection can be conducted varies greatly depending on the combination of software and hardware used, and finding the optimal combination for practical usage in remote areas can be challenging. Internet and remote server access is often limited in these areas, which means that it can be necessary to run detection locally on a less powerful laptop computer. This presents a challenge for performing object detection in a reasonable time frame.

This thesis project examines the possibilities for combining all these technologies in order to create a fully automated system for locating sheep, which can be spread over a large geographic area. It looks at the usage of drones equipped with both thermal and visual cameras to search for sheep, and the usage of object detection technologies on the resulting images in order to find them.

1.1 Research Questions

In order to examine these possibilities, this thesis proposes the following overall Research Question:

Research Question: What is an optimal way to create a system to detect sheep in the field?

In order to support this question, the following sub-questions emerge:

1. What is an optimal way to use object detection to detect sheep in images?
   (a) Which tested object detection configuration is optimal for both accuracy and performance?
   (b) How often can at least one sheep in a group be found?
   (c) How can a combination of thermal and visual images improve object detection accuracy?
2. What is an optimal way to conduct drone flights to detect sheep in nature?
   (a) What is an optimal altitude for the drone flights?
   (b) What is an optimal way to place waypoints for drone images?
3. How can different hardware configurations affect run time and usability of object detection?
   (a) How quickly can detection be performed on different hardware configurations?
   (b) What are the differences between running detection locally and on a remote server?

Chapter 2

Literature Review

This chapter will cover several research papers which are relevant to the writing of this thesis, and how their findings affect the work done during the writing process.

2.1 Previous Master's Thesis

A similar problem regarding locating sheep using a drone was researched in a previous master's thesis from NTNU by J. H. Maribø [3]. This thesis focused on using YOLOv3, a state-of-the-art object detection system, in drone images to find sheep. One of the main conclusions reached was that it is better to class all the sheep in the data set as a single class, rather than to use separate subclasses for different-colored sheep.

The author also experimented with using different detection resolutions and found that increasing the detection resolution did not increase detection accuracy. The author did not, however, retrain his neural network at a higher resolution, but only increased the input resolution. He speculated that this might be why he did not achieve better results with increased resolution.

The author also suggested that the use of thermal imagery and higher training resolutions could improve overall detection accuracy, which is a subject that has been explored in this thesis.

2.2 Methods for Tracking Sheep

In a master's thesis from 2019, L. Svendsen discusses different alternatives that are being used today, utilizing radio and GPS signals [12]. The author mentions several companies offering collars with a radio chip attached for the sheep to use. Just like in cell phones, weak signal reception is common in outlying areas. Most of the collars use ground-based networks, and one of them uses satellites. The latter has better coverage in the terrain but is also more expensive.

The author lists the prices for each device, ranging from 750 to 1890 NOK plus an extra amount for subscription to the service. If a sheep lives for five seasons using a ground-based signal, the cost will be equal to around 50% of the sheep's value. Using a satellite signal, it is almost 100%. There is disagreement on whether all sheep should be radio-tracked, or if only some of them should. For example, tracking only half of the sheep will cut the cost of tracking devices in half, but may also result in more missing sheep.

However, sheep are highly social animals that usually move in groups [7], so it is often enough to locate only a few of the animals. If one animal in a group can be located, the rest are usually nearby. On the other hand, if none of the sheep in a group have been marked with tracking technology, the entire group will be impossible to locate using this method. The principle of locating at least one sheep in a group can also be used in detection using drone images, as mentioned in Research Question 1b.

2.3 Thermal Animal Detection

Burke et al. worked to find the optimal approach for detecting animals using drone-based thermal cameras [13]. They analyzed some of the core problems involved with this type of detection and found that given the right conditions for detection, it can work well.

Some of the main findings were that ground temperature has a huge impact on success rates in detecting animals from the air, and that the resolution of the thermal imaging sensor used has a large impact on how high the drone can be flown and still successfully detect them. They also found that atmospheric conditions can have an impact on thermal imagery, but that this effect is small when the images are taken facing straight down. It can have more of an impact on images which are taken at an angle.

In general, this type of detection can work best if there is a sufficient difference in temperature between the animals and the surrounding terrain, and if the sensor has enough resolution for the target to be noticeable at the chosen flight altitude. They argue that one should carefully analyze local climatic conditions in order to choose the date and time of day when the ground temperature will be at its lowest.

Cukor et al. researched ways to protect roe deer fawns in the agricultural landscape [14].

Thousands of wild animals, including the roe deer fawns, were killed each year due to spring harvesting using heavy machinery. Even though other protection techniques significantly reduced the fatalities among the animals, they were rarely used because they required too much planning, which delayed the agricultural operations.

The authors came up with the idea of using an unmanned aerial vehicle (UAV) with thermal cameras and object detection. To get the largest possible temperature contrast between the fawns and its surroundings, the test flights were conducted during early morning hours. Their results gave a 100% detection rate of the fawns when the camera was 50-70 meters above the ground. Harvesting could be done immediately after detecting the fawns and moving them to a safe place. This improves the efficiency of the whole process, which also increases the chance of farmers taking the time to move the fawns.

2.4 Visual Animal Detection

Van Gemert et al. worked on using drones and computer vision techniques to detect and count animals, with the goal being to analyze whether this technique could be used for conservation efforts, where counting animal populations is very important [15].

In order to evaluate techniques, they constructed a data set consisting of images of cows in fields and used computer vision to attempt to count and localize them. They did not use a neural network-based approach, because they found that such methods cannot be run locally on the drone itself.

In the article, they evaluated three different traditional computer vision methods: DPM (Deformable Part Model) [16], Color-DPM [17], and Exemplar SVM (Support Vector Machine) [18]. They expected DPM to be the better-performing method, as indicated by the literature they reviewed. However, they found that Exemplar SVM gave a surprisingly high average precision score of 66%, which they credit to this method being better suited for small-scale objects than DPM and limited variation between the test and training data set.

The main conclusion was that reported precision scores on ”human-scale” imagery, taken relatively close to the ground, do not necessarily represent how accurately one can detect small animals from high altitudes.

2.5 Object Detection with YOLO

Redmon et al. introduced a new approach to object detection in 2015 in their YOLO paper [19]. They describe the new approach as framing object detection as a regression problem to spatially separated bounding boxes and associated class probabilities, with a single neural network predicting everything in one evaluation of the image. At the time, this was

the fastest general-purpose object detector in the literature and pushed the state-of-the-art in real-time object detection. However, object detection evolves quickly, and Redmon introduced YOLO9000 in 2016 [20], and YOLOv3 two years later in 2018 [21].

YOLOv3 is the current state-of-the-art for real-time object detection and performs very well on accuracy. Joseph Redmon et al. developed a network architecture specifically for YOLOv3, Darknet-53, which significantly improved the detection accuracy on smaller objects within the image. The paper [21] shows the results that Redmon came up with when comparing YOLOv3 to other state-of-the-art algorithms.

2.6 Summary

Svendsen researched different alternatives for tracking sheep using radio and GPS signals. He found that the costs involved in this type of animal tracking can be prohibitive for mass usage, depending on the number of sheep which need to be tracked. He also mentioned the uncertainty surrounding how many of the sheep that need to be tracked for this method to be effective in finding every animal.

Sheep usually move in flocks, so it could be sufficient to locate one animal in a group in order to find every sheep. However, using tracking beacons on a small subset of the animals means that there might be groups where none of the sheep are tracked. Using drone-based computer vision to locate the sheep could mean significantly lower costs, as only one drone is needed. It also avoids the problems with only some of the sheep being tracked.

Several previous authors have attempted both thermal and visual methods of tracking animals in the wild using computer vision techniques. Van Gemert et al. found that detecting small animals from high altitudes can often give very different results to the reported accuracy numbers for computer vision algorithms. Certain algorithms can have advantages and disadvantages in this type of detection which do not manifest in ”human-scale” object detection.

Burke et al. found several factors that can affect results when using thermal detection, including altitude, ambient temperature, and thermal camera resolution. These factors mean that thermal detection should work best at lower altitudes, and at night or during the winter when ambient temperatures are low.

Maribø found in his master's thesis that YOLOv3 is a state-of-the-art real-time object detection system. It achieves a high level of accuracy, and he suggested that utilizing higher resolution images could lead to even better results. This was the impetus for the technique of splitting the input images into several smaller parts in order to preserve more of the original detail, as discussed in section 5.1.1. He also found that it is better to class each sheep as the same class, rather than divide them into subclasses based on color.

Chapter 3

Theory

This chapter will explain and discuss several theoretical topics that are relevant to this thesis project, including many of the technologies and techniques used.

3.1 Drone Technology

A *drone* is a small, unmanned, semi-autonomous vehicle. They were initially designed for military applications and are still often associated with this area. In recent years the use of small civilian drones has grown into a substantial global market [1], and these small, affordable drones open up many new possibilities that were previously restricted to helicopters or airplanes [22].

Civilian drones are usually either of the fixed-wing or rotary-wing type. These rotary-wing drones are usually either of the twin- or quad-copter types, and there are also some designed to function as a traditional helicopter, using a single main rotor for lift and a smaller tail rotor for yaw control [22].

Fixed-wing drones are much more efficient aerodynamically and are capable of flying much longer distances than rotary-wing drones. They can also be designed with gasoline engines, which can hugely increase their flight time. Rotary-wing drones generally require much more precise control over rotor speed to maintain control, which precludes the use of non-electric motors. They use thrust from their rotors to stay in the air, which means that they cannot achieve the same extended flight time as a fixed-wing drone.

Quad-copter drones have become a popular design in recent years. This design incorporates four separate propellers, which means that the drone can be controlled by simple differential power to the rotors, and all the power can be used for lifting purposes. How-

ever, they require four electric motors and four propellers, which can increase the cost, complexity, and weight of the drone.

Smaller rotors are also aerodynamically less efficient, due to the limitations of the decreasing Reynolds number causing an inherent reduction in the lift-to-drag ratio of the rotor at smaller scales [22]. This means that drone with a single large rotor is inherently more efficient than a multi-rotor drone.

By comparison, a twin-rotor design can achieve pitch (up-down) and yaw (left-right) control by differential thrust, but it requires another way to roll the vehicle. A drone designed like a "traditional" (tail-rotor) helicopter cannot control pitch or roll by differential thrust, and the tail rotor consumes an amount of energy simply to control the yaw of the drone.

Both these designs can achieve control by using differential pitch control of the blades (like a traditional helicopter), aerodynamic control surfaces, and thrust vectoring. All these methods are significantly more complex than differentially changing the speed of the motors.

Drones navigate by a combination of GPS, gyroscopes, and altitude sensors, and they also often use visual cameras to aid in small-scale (low-altitude) navigation. In Norway, an unlicensed drone is allowed to fly up to 120 meters above ground level, and is not permitted to move outside the operator's view. It is also not allowed closer than 150 meters from buildings, people, or traffic [9]. As of July 2020, both private and commercial use of a drone weighing between 250g and 4kg will require passing an online course and exam [23].

For the development of this thesis, the authors had access to a *DJI Mavic 2 Enterprise Dual* quad-copter drone. This drone weighs 899 grams, can fly for about 30 minutes, and is equipped with both thermal and visual cameras [24].

3.2 Camera Technology and Nomenclature

This section covers essential terms that are commonly used when discussing camera technology.

3.2.1 Field of View

The *field of view* (FOV) of a camera represents how large an angle the camera can see at any given moment. It is typically represented as an angular FOV, where the angle represents how wide the vision cone of the camera is. The distance from the subject then determines how large an area can be seen in the image, and how large objects in the image will appear, as shown in Figure 3.1.

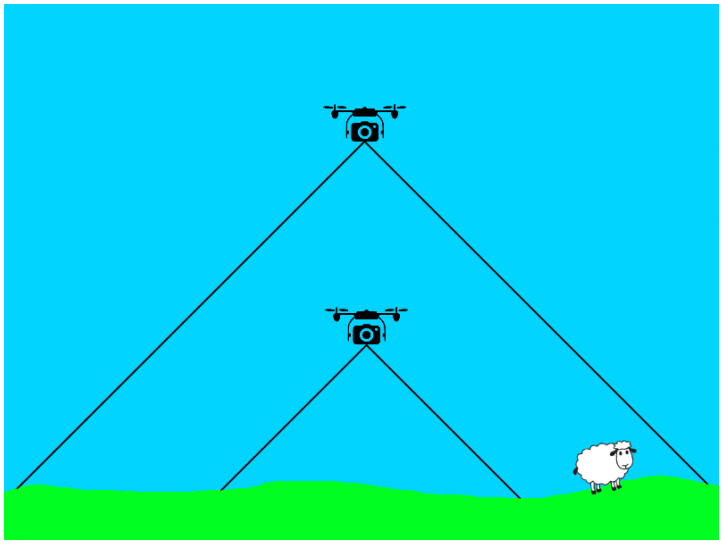


Figure 3.1: Example of camera FOV affecting visual area.

3.2.2 Resolution

A digital camera has a set *resolution*, which is the number of pixels in the X and Y axis that make up a complete image. A digital image consists of a 2-dimensional array of pixel values, which can be stored and transmitted as digital information and reconstructed to show the original image.

The FOV and resolution of a camera determine how many pixels-per-degree the resulting image has. If the subject is very far away, smaller objects will consist of a lower number of pixels, which makes them more challenging to recognize in the image.

3.2.3 Thermal Imaging

Objects emit light in the infrared spectrum, with an intensity depending on their temperature [13]. A camera that can detect this light is known as a thermal camera, and they are often used to detect living creatures in images. Figure 3.2 shows an example of thermal imagery, where sheep can be seen against the background in the infrared spectrum because they are warmer than the background terrain.

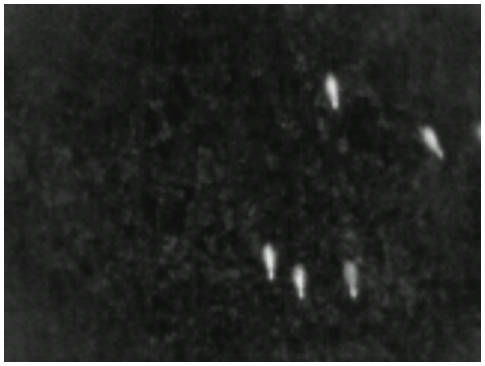


Figure 3.2: Example of sheep in thermal imagery.

3.3 Artificial Neural Networks

The human brain processes information using a network of neurons, each of which performs a simple task and communicates with all the others. In recognizing a human face, each neuron takes a few milliseconds to process the information, while it takes the brain as a whole a few hundred milliseconds. This means that the brain uses about 100 iterations to recognize a human face, compared to a traditional computer vision algorithm which can take thousands and thousands of steps to attempt the same task [25].

An artificial neural network is a computer system that was originally developed to simulate the functioning of the brain. Whereas a traditional computer program is only capable of following an extremely rigid set of instructions and processing carefully formatted data, an ANN is an attempt to create a system that can process information more like the biological network of neurons in the human brain.

In modern computer vision applications, Convolutional Neural Networks (CNN) are the most popular form of neural networks. They were introduced into the ImageNet Large Scale Visual Recognition Competition (ILSVRC) in 2012, and have since grown in popularity, especially after 2014 when R-CNN (Regions with CNN Features) [26] achieved a breakthrough in performance [10], as shown in Figure 3.6.

CNNs were designed to emulate the visual cortex of an animal brain, and are used for processing data which consists of a grid pattern. It is especially useful for detecting and extracting features in images, which consist of two-dimensional arrays of pixels [27]. These features include colors, edges, and patterns.

The CNN uses layers to progressively extract more and more complex features from the image data. These layers are of several different types.

Convolution layers perform feature extraction by applying a kernel (a matrix of numbers)

to the input in order to obtain the element-wise product of the input and the kernel. This product is the output value at each location, creating a map representing the features of the image. Figure 3.3 shows an example of convolution. The training process for a CNN mainly consists of learning which kernel to use in order to reach the output that is closest to the ground truth, which is known from the human-labeled training data.

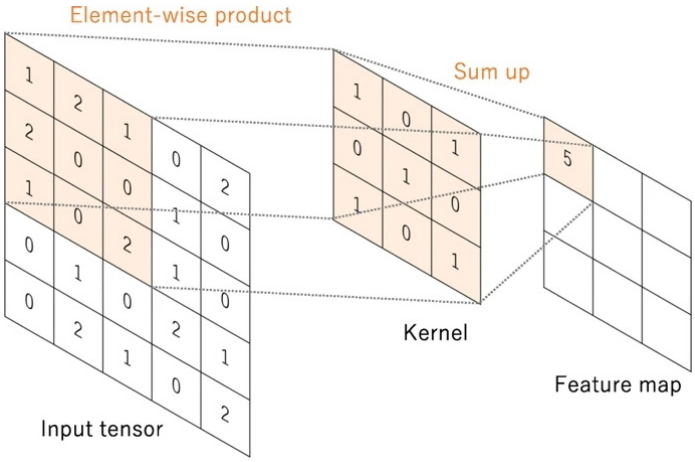


Figure 3.3: Convolutional layer feature extraction [27].

Pooling layers are used to downsample the feature maps created by the convolution layer. This is done to reduce the size of the output, in order to reduce the number of learnable parameters and memory footprint of the map. The most common form of downsampling used is max pooling, where the maximum value is selected, though it is also possible to use other methods like average pooling. Figure 3.4 shows an example of max pooling, where the output is simply the highest value from the four input numbers.

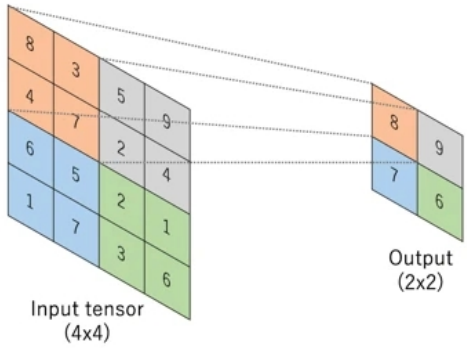


Figure 3.4: Max pooling [27].

Fully Connected Layers are responsible for the actual object detection in the neural network. They map the features to the final output of the network, which in object detection consists of the locations of objects in the image.

3.4 Object Detection

Object detection involves the localization and classification of objects within images. An object detection algorithm typically attempts to recognize the position of learned objects within an image and create bounding boxes around any found objects. Figure 3.5 shows an example of object detection on sheep in a field.

Figure 3.5: Example of object detection on sheep.

The use of machine learning and neural networks has meant that the field of object detection in images has made huge strides in recent years. In 2014 the first modern machine-learning-based object detection algorithm, R-CNN, was developed. It was a huge leap forward for the field of object detection, which had largely plateaued with the use of ”traditional” detectors since 2010 [10].

The pinnacle of traditional object detection, ”Deformable Part-based Model” (DPM), had been introduced in 2008, and the final version of it had achieved an mAP score of 33.7% on the VOC07 data set. The first version of R-CNN managed to improve the mAP to 58.5%, which set off an explosion of progress in the use of neural networks for object detection. Figure 3.6 shows an overview of advances in object detection performance.

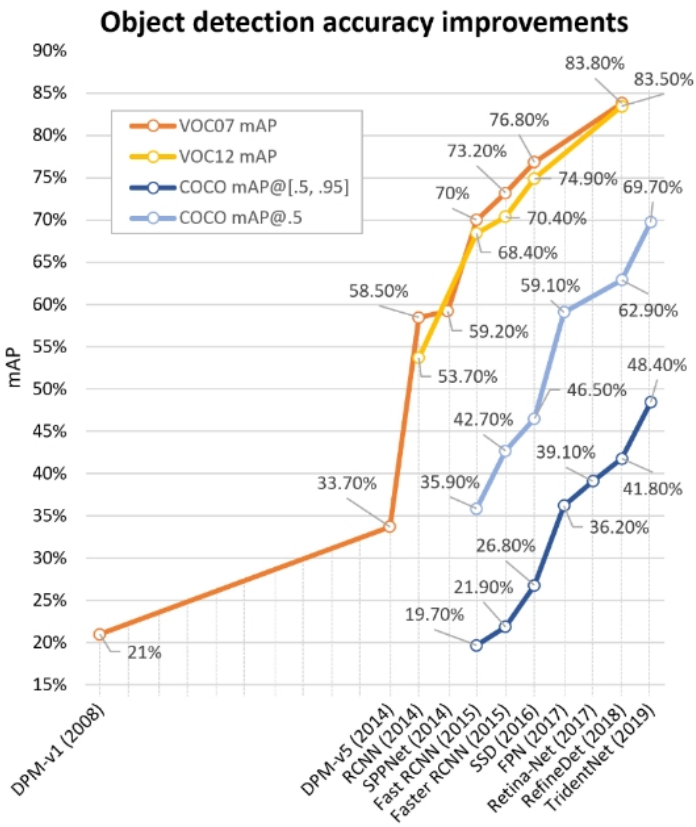


Figure 3.6: Historical improvements in Object Detection on VOC07, VOC12, and COCO data sets [10].

While R-CNN had much better detection results than traditional methods, it was still quite slow, taking 14 seconds per image on a GPU-accelerated machine. R-CNN is a multi-stage detector, which uses several passes to detect objects in the image. In 2018 YOLOv3 was released, a single-stage detector that had similar performance to other networks but at a much faster speed [21]. It is capable of running detection in real-time, and was chosen for this project for being a fast and accurate state-of-the-art detector.

3.5 You Only Look Once

There are two common approaches when it comes to object detection. The difference between them is how the algorithms interpret images.

The first one is based on classification, such as R-CNN. These algorithms select interesting

regions in the image and classify them. When all the regions are classified, the algorithm has to run again. Repeating this process over and over slows down the detection speed. The other approach predicts classes and bounding boxes after just one run of the algorithm. This process is fast and is commonly used for real-time object detection. This is how YOLO (You Only Look Once) does it and the reason it became so popular.

The first thing YOLOv3 does is dividing the image into an SxS grid of cells. Each cell is responsible for predicting objects inside it using bounding boxes. These boxes are estimated using K-means and predefined anchor boxes. By default, there are nine anchors, three for each layer.

The anchor boxes are height-to-width ratios with a size that is common for the objects in the data set. The bounding boxes have five components: x, y, w, h, and confidence. The x and y components are the coordinates of the center (relative to the grid cell), and the w and h components are the dimensions (also relative to the grid cell).

The last component, confidence, is how certain the prediction of an object is. YOLO uses three different scales to predict objects with a grid stride of 32, 16 and 8. Each of these scales will need to handle a large number of bounding boxes. If the input image is 832x832, YOLO makes detections on the scales 26x26, 52x52, and 104x104, with three predictions on each scale [28]. In this example, the total number of bounding boxes will be:

$$((832/32)^2 + (832/16)^2 + (832/8)^2) * 3 = 42588$$

or formulated differently:

$$((26 * 26) + (52 * 52) + (104 * 104)) * 3 = 42588$$

These boxes are pruned using thresholding by objectness and non-maximum suppression. This means that overlapping bounding boxes with a confidence score higher than the threshold get pruned into only one box with the object inside of it. The threshold is set to 0.25 by default but can be modified to any desired number [29].

3.6 YOLOv3-tiny

The creators of YOLO have developed a new variation of YOLOv3 called YOLOv3-tiny. This algorithm runs at a much higher speed than its big brother but at the cost of lower accuracy. It has an mAP of 23.7% on the COCO benchmark data set, which is less than half that of YOLOv3's score of 57.9%. It is approximately 442% faster and can achieve up to 244 FPS on a single GPU [30]. The weight file generated by YOLOv3-tiny was around 35MB, while YOLOv3's weight file was around 235MB. This reflects the simplicity of the tiny version.

3.7 Darknet-53

The second version of YOLO, YOLO9000, has problems detecting smaller objects. YOLO9000 uses a network architecture called Darknet-19. This consists of 19 layers, and YOLO9000 used 11 layers for detection. In total, there are 30 layers.

YOLOv3 uses Darknet-53, which is illustrated in Figure 3.7. As the name suggests, it has 53 layers and is trained on ImageNet, an online database for images [31]. To improve detection even further, YOLOv3 uses another 53 layers making a total of a 106 layer fully convolutional underlying architecture [32].

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3.7: The architecture of Darknet-53 [21].

This network is much more powerful than Darknet-19 and is still more efficient than other commonly used networks like ResNet-101 or ResNet-152. The official YOLOv3 paper provides these results when testing on ImageNet using the same variables for all of them:

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	**171**
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

Figure 3.8: Comparison of backbones. Accuracy, billions of operations, billion floating-point operations per second, and FPS for various networks [21]. This shows that Darknet has significantly higher FPS than other backbones.

3.8 General Performance Metrics for Object Detection

Several metrics are generally used to measure the performance of an object detection system. These metrics are measured by running the system on a test data set which has been labeled by humans, and comparing the results to the known truth. These are also used for explaining the results in this master's thesis.

3.8.1 Intersection Over Union

Intersection over Union is an evaluation metric used to measure the accuracy of the detection of an object. Any algorithm that provides predicted bounding boxes as output can be evaluated using IoU [33].

IoU essentially compares the overlap between the bounding boxes created by the detection system and the known locations of pre-labeled objects in the testing data set to evaluate the accuracy of object detection. This is done by comparing the ground-truth bounding boxes (predefined boxes around objects in the image) with the predicted output bounding boxes. The larger the IoU is between these two, the higher the score it gets. It is calculated with the formula:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

Figure 3.9 illustrates how IoU is calculated.

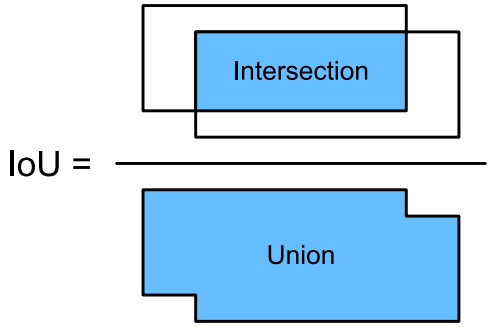


Figure 3.9: Illustration of Intersection over Union.

By setting a minimum threshold for the IoU, this metric can be used to automatically evaluate whether a particular detected object is a true positive or false positive. This is then used to calculate other performance metrics.

3.8.2 Recall

Recall is a measure of how many of the actual objects in the image are detected. It is calculated by the formula

$$\text{Recall} = \frac{TP}{TP + FN}$$

where TP is ”True Positives” and FN is ”False Negatives”, or actual objects in the image which were not detected by the algorithm.

3.8.3 Precision

Precision measures how many of the detected objects are true positives (as opposed to false positives). It is calculated with the formula

$$\text{Precision} = \frac{TP}{TP + FP}$$

where FP is ”False Positives”, or detected objects which are not actual objects.

3.8.4 Mean Average Precision

Increasing the precision of the detection algorithm means decreasing the recall, and increasing the recall reduces precision. This is because detecting more true positives in the image (increasing recall) always leads to more false positives and vice versa. This trade-off can be shown on a precision-recall curve, and *Mean Average Precision* (mAP) is a measure of the average value of this curve. It is a very common value for measuring the overall quality of a detection algorithm.

Chapter 4

Platform

This chapter will explain the complete solution that was developed, both as a proof-of-concept and to test the capabilities of such a system.

4.1 Client-Server Architecture

Client-server architecture is a computing model in which the server hosts, delivers, and manages most of the resources and services to be consumed by the client. This type of architecture has one or more client computers connected to a central server over a network or internet connection [34].

Figure 4.1 illustrates the architecture used for the developed system. The drone sends and receives data between itself and the client, and the client handles the data from the drone and relays it to the server. The server will respond with results when it is finished processing the data from the client.

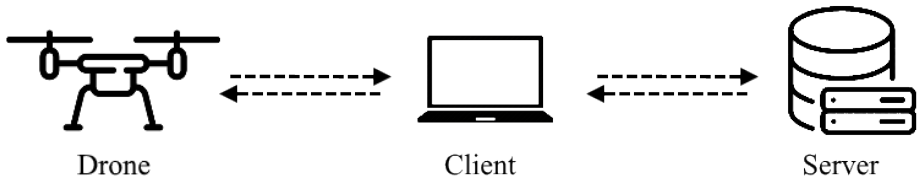


Figure 4.1: Illustration of client-server architecture with a drone.

The system was developed in a manner that makes it relatively simple to deploy the server

solution on any machine, including the same machine which is running the client. This means that the system can also be run as a self-contained unit, with one machine running both the client software and the image processing system.

4.2 Client-Side

The client-side of the platform was developed as a Universal Windows Platform (UWP) app in C#. It was made as a UWP app because of the seamless integration with the DJI Developer SDK, which is used to develop applications for DJI drones. The application allows the user to select an area on a map and view the flight path of the drone. Once the user is satisfied with the path of the drone, they can start the search. The flight altitude is modifiable between 15 and 120 meters, to allow for larger areas to be searched at the cost of less accurate object detection.

The flight path is calculated and generated as a grid pattern of waypoints within the rectangular, user-selected area on the map. The distance between waypoints is determined by the combination of the field of view (FOV) of the drone camera and the selected flight altitude. The drone used while developing the system, DJI Mavic 2 Enterprise Dual, has two cameras with different FOV. The visual camera has a FOV of approximately 85°, while the thermal camera only has a FOV of 57° [24]. The lowest FOV, 57°, was used to calculate the maximum distance between waypoints. This ensures full coverage of the selected area for both the visual and thermal camera, as illustrated in Figure 4.2.

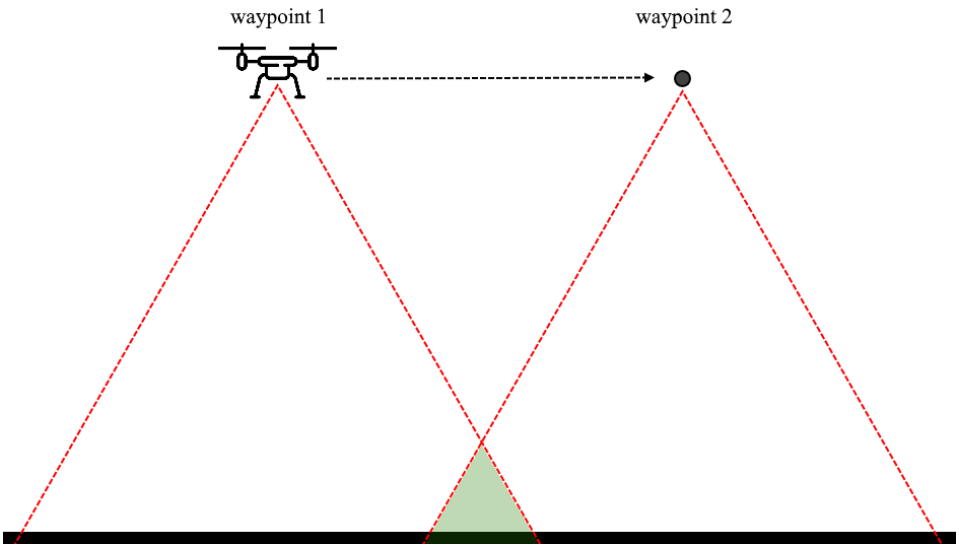


Figure 4.2: Illustration of image overlap between waypoints.

Once the user starts the mission, the flight path data is transferred to the drone, and then it takes off. The application shows the current location and altitude of the drone and the current status of each waypoint. The drone will then automatically fly through the provided path and stop momentarily to take images on each specified waypoint. The drone stops on each waypoint before taking a photo to ensure the best possible image quality. A better camera or lower shutter speed would possibly allow for taking images without stopping momentarily, which would prolong flight time. Once the flight path is finished, the drone will return to where it took off.

The images are then sent to the server for object detection. The user has to input the IP address and port to the object detection server to specify for the program where to send the images; this will require a stable internet connection. It is possible to run the server locally if no internet connection can be established in the field. If an object is detected in any of the images, the application will update the status of the corresponding waypoint.

The user can click on a waypoint to see the image combined with the object detection data from the server in order to verify the results. The application will overlay the position of detected sheep in both thermal and visual images onto the visual image, and the user can also click a button to show only the thermal image results. If the thermal and visual detections overlap more than a set percentage, the result will be shown as a single detection with a different color from the visual or thermal detections. Figure 4.3 shows a simulated example of how this works. The left side shows visual detection results, the right side shows thermal detection results, and the middle picture shows the resultant application UI

display, with the overlapping detection colored yellow.

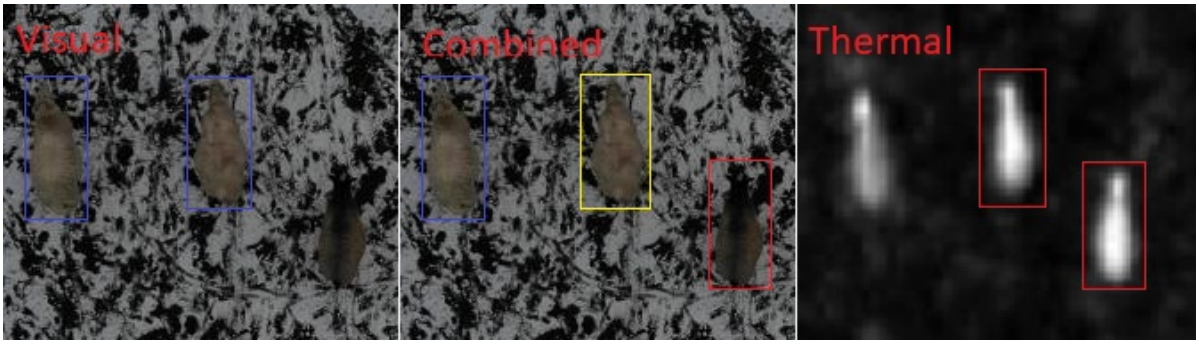


Figure 4.3: Illustration of detection UI.

Figure 4.4 shows the drone during a simulated flight at an altitude of 75 meters. The drone takes pictures of a printed-out photo of sheep to imitate a real flight. The waypoints have different color codes to show their current status. Blue is the default color code, and it indicates that the drone has not taken a picture at this waypoint yet. Green indicates that sheep were detected, yellow indicates that the image is still being processed, and red indicates that no sheep were detected. The user can select a waypoint to view the prediction in real-time and toggle between the visual and thermal images, as shown in Figure 4.4.

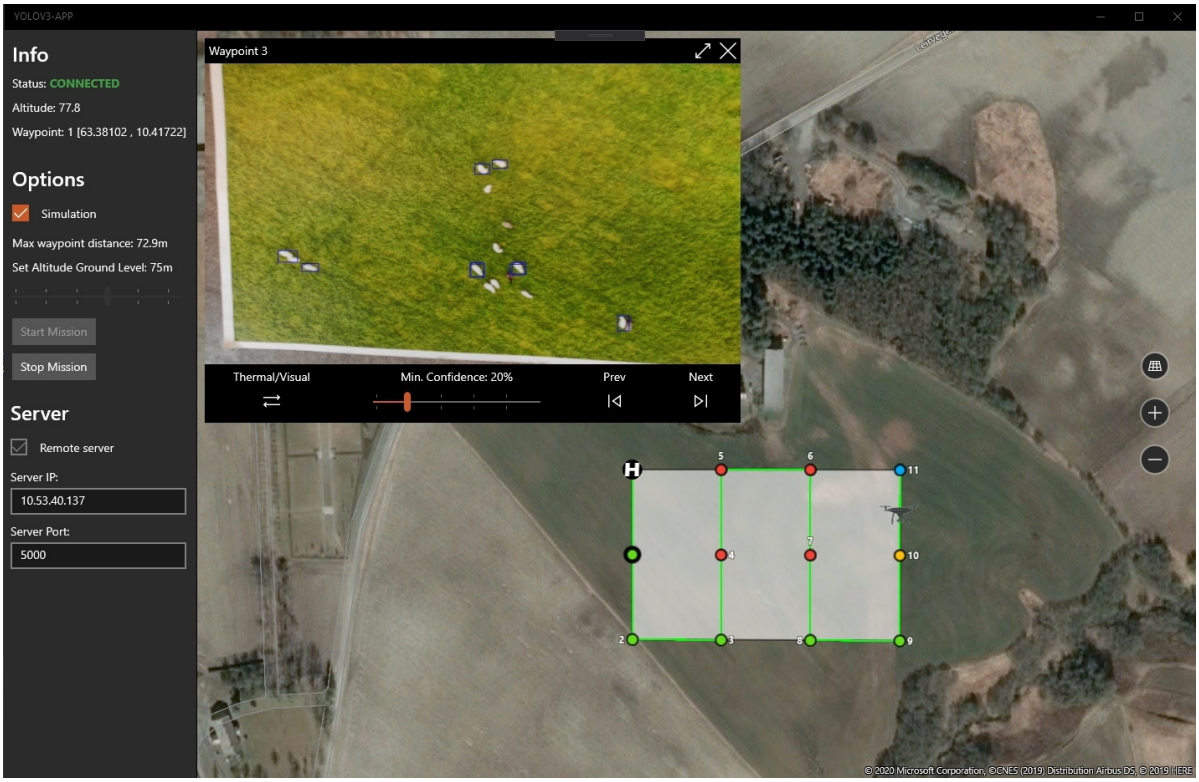


Figure 4.4: UWP application during a simulated drone flight.

4.3 Server-Side

The server-side of the platform was developed as a simple REST server, using the Python framework Flask to handle HTTP requests. It handles incoming requests, in the form of drone images from the client. They are passed to a compiled version of YOLOv3, and the server then sends the corresponding detection data back to the client.

The server-side solution was developed to allow for easy use of differently compiled versions of YOLOv3 and to allow for effortless substitution of the different types of detection algorithms. This meant that it was easy to test the performance of the system with different combinations of YOLO configurations and hardware, and compare the advantages and disadvantages of running the detection on these.

Chapter 5

Method

This chapter will cover the methods used during the work performed for this thesis, including data set acquisition and testing methods. It will also cover a number of statistics about the data set and methods.

5.1 Data Set

Testing thermal and visual image recognition requires at least one data set for both types of images. The images used for this project were provided by the supervisor and were labeled using LabelBox [4], an online collaboration tool for labeling images.

5.1.1 Visual Images

Image Resolution and Splitting Images

Before detecting objects in an image, YOLOv3 will resize the image to whatever input resolution it is configured to use. The effective maximum resolution it could be trained at before running out of memory on the available hardware was 1024x1024 px. This is a significantly lower resolution than the drone images used in this system. Running YOLOv3 directly on these images would lead to a substantial loss of information from the input data, as each sheep in an image would potentially consist of a small number of pixels. It was therefore decided to explore whether splitting the images apart into a grid of input images could improve the performance of the object detection algorithm.

Visual Data Set

The data set used for training and validation consists of 341 visual images. These images were split using a script with an overlap so that no sheep would get split in two, as shown in Figure 5.1. After splitting, the data set consisted of 7952 visual images.

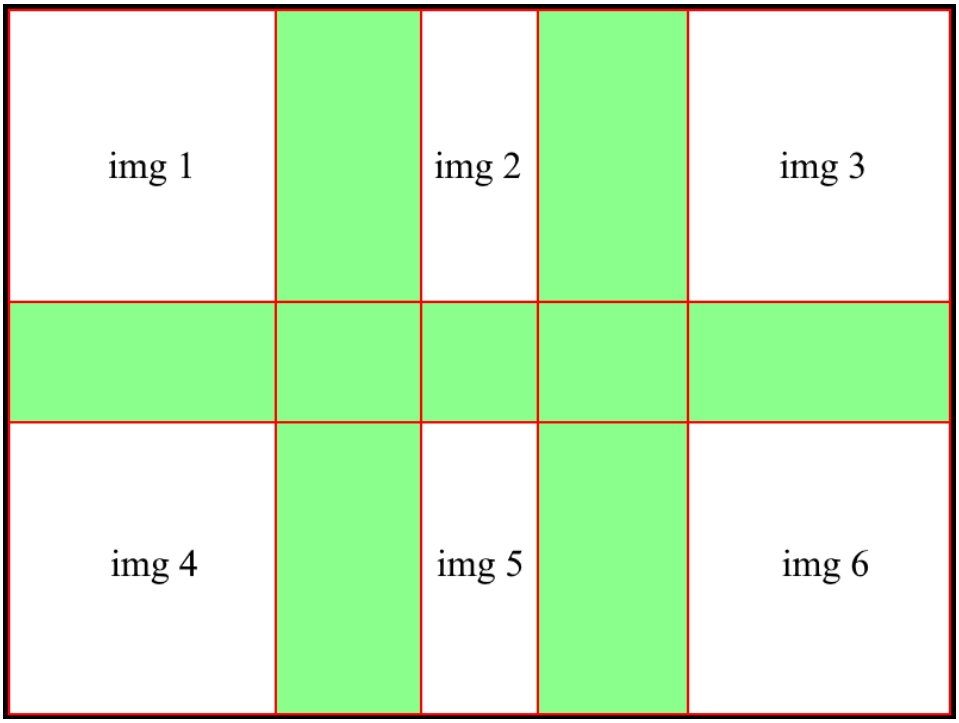


Figure 5.1: Illustration of the splitting process. The original image is split into multiple, smaller images. The overlapping areas are marked green.

The images were stitched back to the original form, and overlapping labels were removed. This resulted in two different visual image data sets that could be used to figure out whether high or low-resolution images gave the best results.

Following are some of the characteristics of the visual image data set:

1. The original resolution of the images is 4056x3040 px or 5280x2970 px, depending on which drone was used. Some of the images in the data set were taken by a DJI Inspire 2 drone, which has a different camera resolution.
2. The resolution after splitting the images is 1024x1024 px.
3. The images are all taken from a drone at a vertical angle downwards.

4. The background in the images varies depending on the weather and natural changes in nature. Some have snow or grass; others contain trees and rocks.
5. The images were taken at different times of the day. Some are taken during the night, which makes them darker.

5.1.2 Thermal Images

The data set used for training and validation consists of 145 thermal images provided by the supervisor. Due to the low resolution in these images, they were not split.

Following are some of the characteristics of the thermal image data set:

1. The resolution of the image files is 640x480 px, which is automatically upsampled by the drone from the sensor resolution of 160x120 px.
2. The images are all taken from a drone at a vertical angle downwards.
3. The image contrast is automatically enhanced by the developed system, so that the highest temperature pixel is white and the lowest temperature is black, as shown in Figure 5.2.
4. The images were taken on different days. Some are taken during late summer and others in late autumn after a snowfall.

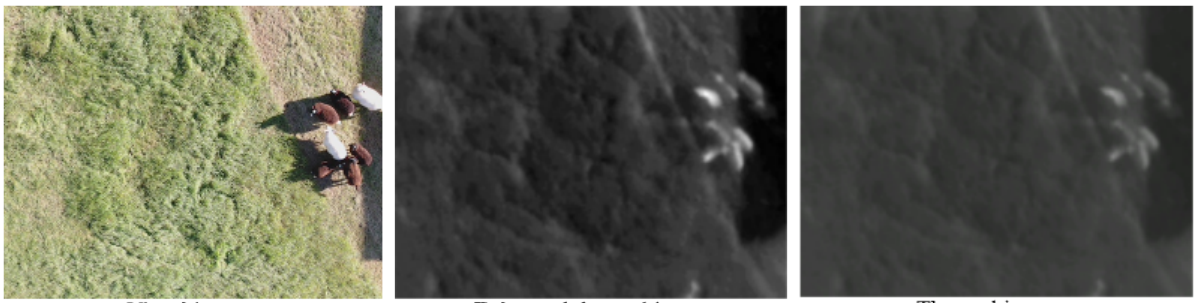


Figure 5.2: Editing the thermal images improved the visibility of the sheep when the temperature was relatively high.

5.2 Labeling

The images were labeled using LabelBox [4]. This labeling tool allowed for collaboration and provided an efficient way to label a large number of images. Each sheep was marked

within a bounding rectangle to provide information for the training phase of YOLOv3, as seen in Figure 5.3. The label data was exported as a CSV file and converted to the correct format needed by YOLOv3.

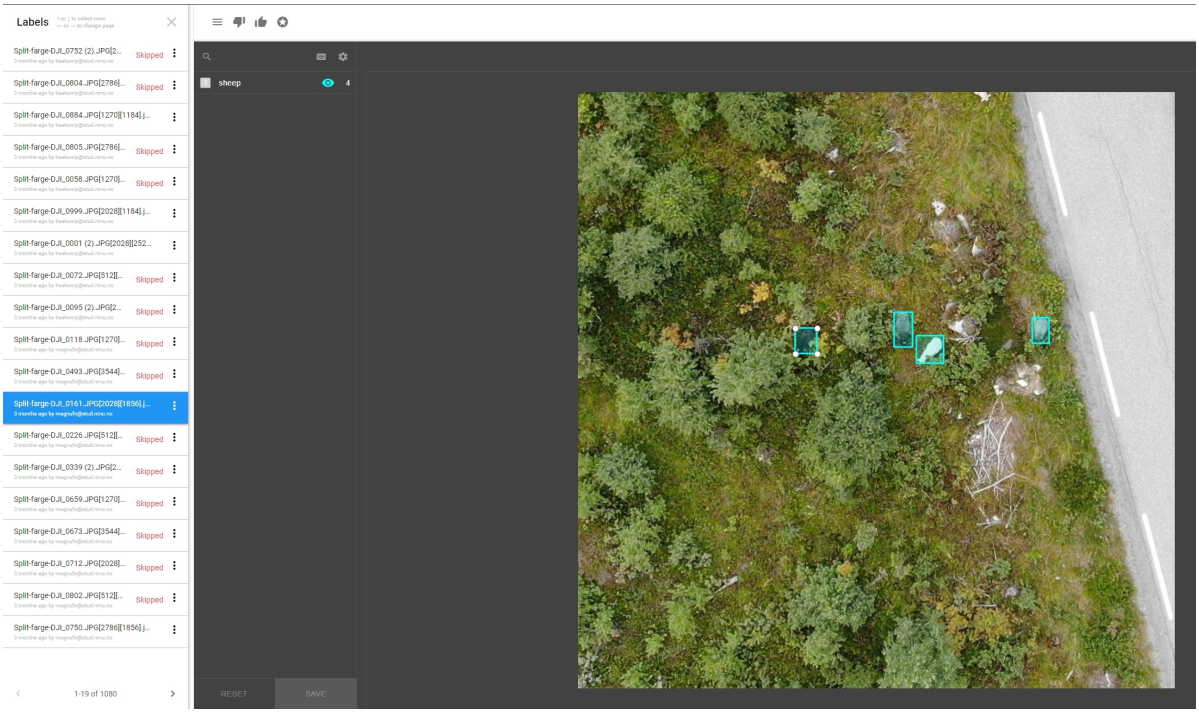


Figure 5.3: LabelBox with multiple sheep marked.

Table 5.1: Distribution of sheep in data sets and total labels.

Data set type:	With sheep:	Without sheep:	Total:	Total labels:
Visual	2801 (35%)	5151 (65%)	7952	9905
Thermal	939 (58%)	676 (42%)	1612	3665

5.3 Training

The network was trained on the IDUN cluster, administered by NTNU's High-Performance Computing group. The cluster consists of multiple high-end processors and graphics cards. The training speed was improved by utilizing graphics cards. The GPUs used, NVIDIA Tesla V100 and P100, are optimized for tasks such as training neural networks. This allowed for rapid training on the different data sets.

Table 5.2: Specs for the IDUN cluster used for training the neural network [35].

Nodes:	Processor:	Processors:	Cores:	Memory:	GPU:	GPUs:
27	Intel Xeon E5-2630 v4	2	20	128GB	-	0
9	Intel Xeon Gold 6132	2	28	192GB	-	0
8	Intel Xeon E5-2695 v4	2	36	128GB	NVIDIA Tesla P100	2
19	Intel Xeon E5-2650 v4	2	24	128GB	NVIDIA Tesla P100	2
5	Intel Xeon Gold 6132	2	28	768GB	NVIDIA Tesla V100	2

The different neural networks were all trained for sufficient amounts of time. The resulting weight files were selected based on the mAP and IoU plots for each network, as shown in Figure 5.4. Due to overfitting, the best results are not always achieved during the last iteration of the training process [36]. As a result of this, the weights were selected where the mAP and IoU scores leveled out, to minimize overfitting.

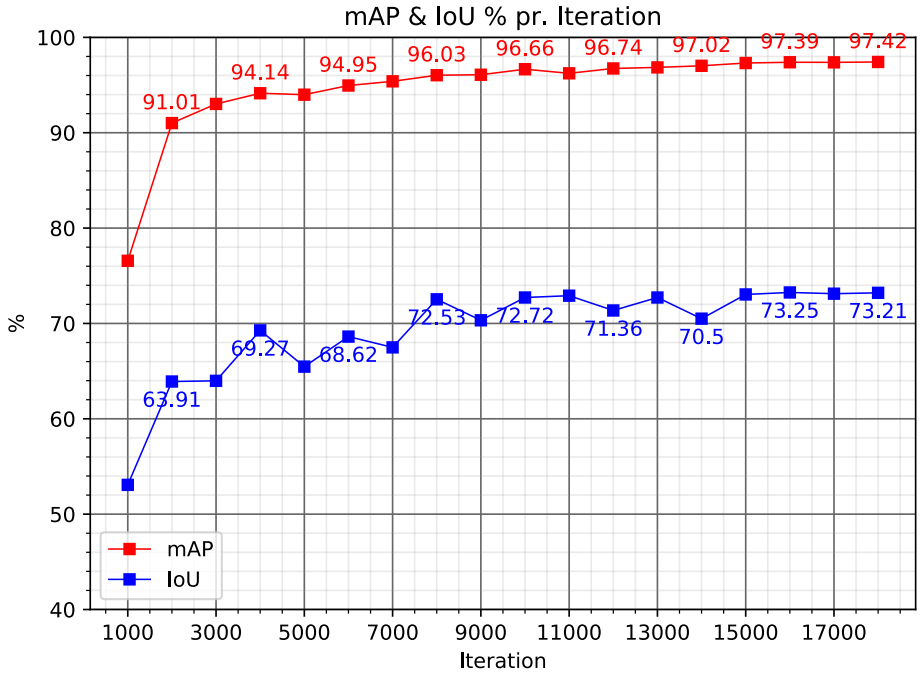


Figure 5.4: YOLOv3-tiny: mAP & IoU % plot for every 1k iterations.

5.4 Testing

Both manual and automatic testing were performed on multiple versions of YOLOv3 with different settings. Initially, it was found that manual testing returned results that were much better than expected. It was inferred that this might be because many of the images used for the manual testing, while not taken directly from the training data set, came from the same series of images that were used in training the algorithm.

It was believed that the images were too similar to the training data set. Mirroring the images horizontally and vertically immediately gave worse results in manual testing, which supports this conclusion. The manipulated images gave results that might be closer to real-life results, and it was decided to use images that had been manipulated in this way to perform comparative testing of the different detection methods being investigated.

A subset of images was manipulated and used to test the object detection algorithms. The test subset was constant throughout the testing. The test images were mirrored both horizontally and vertically to ensure that it was different from the training set (Figure 5.5).

Figure 5.5: Example of a mirrored image used in manual testing.

Chapter 6

Results

This chapter will cover the relevant results of the work which was performed while developing this thesis project. It will include an in-depth comparison of the performance of the object detection system under varying conditions and configurations, and an analysis of how they affect detection accuracy and run times.

6.1 Accuracy

Maribø had previously explored the alternatives for classifying the sheep in the data set, and found that it is better to class them all as a single object type rather than separating the different-colored sheep [3]. These two alternatives were both initially tested for this thesis, but early results concurred with Maribø's thesis and showed a significantly higher mAP for the single-class approach. The three-class approach was therefore quickly discarded, and the system used for this project was trained using a single class for all the sheep.

However, YOLOv3 will then only report a single mAP number for all the objects in the data set. When they are all classified as the same object type, there is no way for it to be aware of the existence of different types of sheep. It is also not capable of evaluating its own performance in different conditions or terrain types within the same data set. This means that there is no way of automatically testing how well the system performs on specific types of sheep or in specific conditions. While this mAP number is a very useful metric for quickly comparing the different methods, it is not sufficiently detailed for this report.

In order to investigate the performance of the object detection on different types of sheep, the system was run with several different configurations of the detection algorithm on a selection of images, and the results were manually analyzed. The tests were also run on

several different processor types, in order to compare run-times between them.

It should be noted that, as shown in Table 6.1, the data set used for manual testing has far fewer black sheep than white or gray ones. Therefore, the percentage numbers for black sheep will be skewed much more by a small difference in performance.

It should also be noted that the images used for manual testing are a subset of images that were selected by the authors to represent a wide variety of conditions, and the absolute numbers reported here may not be the same as the results found in real use. These tests are intended as comparative tests between the different approaches, and not absolute numbers for the real-life accuracy of the system.

Table 6.1: Number of sheep in the manual test data set.

White:	Gray:	Black:
487	423	127

6.1.1 Split vs. Non-Split Images

In order to compare the effectiveness of splitting the images vs. not splitting them, several detection models were trained on the same data set. Table 6.2 shows these models. Three similarly performing detection thresholds for both split and non-split detection were then selected for comparison. Table 6.3 shows manual testing results for these models.

Table 6.2: Comparison of detection methods.

YOLO version:	Image type:	Iterations:	mAP:
YOLOv3	Split	5K	98.37%
YOLOv3	Non-split	4K	85.20%
YOLOv3-tiny	Split	10K	96.66%

Table 6.3: Manual testing results.

YOLO version:	Method:	Threshold:	White:	Gray:	Black:	Avg. false positives:
YOLOv3	Split	20%	95.5%	89.6%	92.1%	0.41
YOLOv3	Non-split	10%	94.3%	88.1%	93.5%	0.45
YOLOv3-tiny	Split	20%	96.5%	89.0%	95.2%	0.64

This shows that the method of splitting the images has a very similar overall accuracy to the non-split method on the test images, but a lower reported mean average precision from YOLO itself. There are certain images where the split detection method is much more accurate, but for many images the accuracy is quite close, and non-split detection is significantly faster.

However, using YOLOv3-tiny on the split images was found to make detection much faster than ”regular” YOLO, and it also avoids the problems with accuracy at high altitudes in the non-split images. Table 6.4 shows a detailed comparison of detection times.

Table 6.4: Comparison of average detection times.

Type	Storage	CPU/GPU	YOLOv3 Split	YOLOv3 Non-Split	YOLOv3-tiny Split
HPC GPU	HDD	NVIDIA Tesla P100	9.19s	5.79s	6.40s
Desktop CPU	SSD	Intel i7-8700 @ 3.20GHz	37.58s	3.04s	6.03s
Desktop CPU	SSD	Intel i7-4790 @ 3.60GHz	62.85s	4.57s	8.51s
Laptop CPU	SSD	Intel i5-4210U @ 2.70GHz	146.76s	9.23s	18.35s

Figure 6.1 shows an example of an image where there is a noticeable difference in detection performance between the two methods. The drone is quite high up in this image, so the lower effective resolution of the non-split detection means that it has trouble picking out the sheep at this altitude. The split detection analyses the images at a much higher effective resolution, and therefore finds the sheep even when they are very small.

Figure 6.1: Split vs. non-split object detection at high altitude, with outlined areas zoomed in.

In most of the test images, the drone is at a lower altitude. In order to further investigate the differences between these two methods, a smaller test data set consisting only of images taken at higher altitudes was created. This was then run through these detection methods, and the results confirm that split detection in most cases performs better than non-split detection at high altitudes, for both YOLOv3 and YOLOv3-tiny. Table 6.5 shows these results. Interestingly, YOLOv3-tiny shows a higher accuracy in this test, but also more false positives.

Table 6.5: YOLOv3: Comparison of split and non-split detection at high altitude.

YOLO version:	Image type:	Threshold:	White:	Gray:	Black:	Avg. false positives:
YOLOv3	Split	20%	59.4%	28.0%	25.0%	0.78
YOLOv3	Non-split	10%	37.5%	20.0%	12.5%	0.55
YOLOv3-tiny	Split	20%	81.3%	64.0%	25.0%	1.00
YOLOv3-tiny	Non-split	10%	40.6%	20.8%	11.1%	2.33

6.1.2 YOLOv3 vs. YOLOv3-tiny

The relative performance of YOLOv3 and YOLOv3-tiny was also investigated. The results show that YOLOv3-tiny often performs very well compared to ”regular” YOLOv3, and it is noticeably faster. Table 6.6 shows this comparison.

Table 6.6: Comparison of YOLOv3 and YOLOv3-tiny on split images.

Version:	mAP:	Threshold	White:	Gray/brown:	Black:	Avg. false positives:
YOLOv3	98.37%	20%	95.5%	89.6%	92.1%	0.41
YOLOv3-tiny	96.60%	20%	96.5%	88.9%	95.2%	0.64

These results confirm that YOLOv3-tiny performs almost identically to ”regular” YOLO on split images, with an mAP score which is only 1.77% lower. Considering the difference in detection time between the two methods, it is clear that using YOLOv3-tiny with split images is an optimal solution. Table 6.7 shows detection times with YOLOv3-tiny on split images.

Table 6.7: Comparison of average detection times for YOLOv3-tiny, split images, 608x608 px.

CPU/GPU	Split detection time
Intel Core i7-8700 CPU @ 3.20GHz	6.07s
Intel Core i7-4790 CPU @ 3.60GHz	8.51s
Intel Core i5-4210U CPU @ 2.70GHz	18.35s

This difference in detection accuracy does not correspond with the accuracy figures reported for the two methods in the literature, where YOLOv3-tiny is expected to show significantly worse results than YOLOv3. However, as discussed in section 2.4, detecting small animals from high altitude is a different problem from most computer vision tasks, and detection accuracy on human-scale imagery does not always correspond to performance in this type of detection.

6.1.3 Resolution Differences

By changing the detection resolution, the results vary both in performance and accuracy. By decreasing the resolution, the performance speed increases, but the accuracy decreases.

The opposite occurs with a resolution increase, with varying success from test results. The results from Table 6.8 were tested on an i7-8700 CPU, and the network was trained at a resolution of 608x608 px. The detection resolution was the only variable changed.

The increase in false positives with a resolution decrease is expected and confirmed by test results. However, the same increase in false positives also occurs with a resolution increase. This is not expected behavior, but may be caused by the fact that the neural network was trained at a resolution of 608x608 px, and performs noticeably better closer to native training resolution.

Table 6.8: YOLOv3-tiny performance and accuracy at different resolutions.

Resolution:	Time:	Accuracy:	Avg. false positives:
416x416	3.50s	81.8%	0.73
608x608	5.68s	88.0%	0.55
832x832	8.76s	91.0%	0.82
1024x1024	13.13s	88.0%	1.00

Lowering the input resolution for the object detection causes loss of detail in the images, as illustrated in Figure 6.2.

Figure 6.2: Illustration of loss of detail in lower detection resolutions.

These results indicate that higher test resolutions do not directly translate into an increase in accuracy or a decrease in false positives, while lowering the test resolution does directly increase the detection speed of the network. Using this knowledge, the network used in the system was both trained and run at a middle ground resolution of 608x608 px, which performs well in both speed and accuracy, while keeping false positives to a minimum.

6.1.4 Detection Threshold

The initial configuration consisted of running the detection on split images, with a threshold of 10% confidence and the model trained for 5000 iterations. Table 6.9 shows the results for this detection model. This model has an average of 0.91 false positives per image, which is quite high. By increasing the detection threshold, it was found that the number of false positives can be reduced while still maintaining a high degree of accuracy.

Table 6.9 shows the false positive rate decrease by increasing the confidence threshold

for object detection, with a minor loss in detection rates for sheep. In general, all tested detection models showed a similar trend, where the number of false positives goes down as the detection threshold is increased.

Table 6.9: Manual testing: Comparison of performance at varying detection thresholds.

Threshold:	White:	Gray/brown:	Black:	Avg. false positives:
10%	96.3%	90.8%	95.2%	0.91
20%	95.5%	89.6%	92.1%	0.41
30%	94.5%	88.0%	91.3%	0.24

These results show that the chosen detection threshold has a considerable influence on the performance of the detection system. At lower thresholds, the system could generate too many false positives to be useful, as the user would have to manually review the images to eliminate these spurious results. At a high threshold, the system could struggle to detect the animals.

This indicates that it could be useful to be able to adjust the detection threshold while locating the sheep, in order to compensate for varying conditions and detection performance. If a large number of false positives appear, it could be useful to increase the threshold in order to eliminate them. Conversely, if the sheep are difficult to locate, it could be useful to lower the threshold in order to find more of the animals. However, as discussed in section 6.4, if the goal is to locate at least one sheep in a flock, the threshold can be set much higher in order to reduce the number of false positives significantly.

6.2 Detection Time

It is clear from the results that the chosen detection method has a significant impact on the time taken to perform the detection. Splitting the images increases the number of images to be processed by over 20 times, depending on the resolution of the input image. This has a similar multiplicative effect on detection time.

As shown in Table 6.4 and Table 6.7, the hardware on which the detection is run has a dramatic impact on detection times.

The fastest models run faster on the CPU-based machine than on the GPU, however the GPU-equipped machine also uses mechanical hard drives for file storage. The CPU machines have solid-state drives, and this could explain why the run time is longer on the GPU-equipped machine when the overhead related to file management becomes more significant than the benefits of running detection on a GPU.

In practice, the hardware therefore decides which detection model can be used. If it is desirable to run the detection locally ”in the field”, it is not practical to run the split detection method which takes several minutes for each image. However, this method only takes a few seconds on a GPU-equipped server, which can process the images remotely.

However, a laptop could feasibly run the split images with YOLOv3-tiny, which takes about 18 seconds per image on an Intel Core i5-4210U CPU @ 2.70GHz. It is possible that this run time could be significantly lower on a more powerful laptop. As shown in Table 6.6 and Table 6.4, YOLOv3-tiny is much faster on average, and has almost the same detection performance as ”regular” YOLOv3.

6.3 Thermal Detection

The thermal images with increased contrast show an mAP score of 96.85% after 4000 training iterations. It is difficult to determine how these results compare with real-life use of the system, given the limited usable data set and lack of opportunities for testing the system.

In order to thoroughly investigate the effectiveness of thermal detection, manual testing was performed on a smaller data set, consisting of thermal images taken under varying conditions. The results of this testing are shown in Table 6.10. As expected from the reported mAP number, the results are quite good, with 92% of the sheep being successfully detected and an average of 0.55 false positives per image.

Some of the images were taken during the winter, when the terrain was covered in snow. As expected, these images show almost perfect performance of the detection algorithm, as the sheep stand out very noticeably from the background. Detection performs much worse

Table 6.10: Performance of thermal detection.

Iterations:	mAP:	Detected sheep:	Avg. false positives:
4K	96.85%	92.0%	0.55

on images taken in higher ambient temperatures, where there are a large number of false positives detected by the system.

The method of increasing contrast on the thermal images was found to be useful in making small temperature differences stand out when detecting the sheep. It matters most when the outside temperature is relatively high, as this is when the sheep are most difficult to pick out from the background temperature and detect in thermal images. However, this is also when visual detection is most successful.

In general, thermal detection is most useful as an addition to the system in conditions when the sheep are difficult to see, and this is often when the lower outside temperature makes thermal detection most effective. By combining the two methods of detection, the system can successfully detect a greater proportion of the sheep, as discussed in section 6.5.

6.4 Detecting at Least One Sheep in an Image

Because sheep usually move in groups, it can often be enough to find only one of the animals. The results were therefore analyzed in order to investigate how often the system detects at least one sheep in a group.

It was found that using the split detection method always detected at least one of the sheep in every image in the test data set, using the tested confidence threshold of up to 30%. Non-split detection had several images where it did not detect any of the sheep, as discussed in subsection 6.1.1.

In order to further investigate this alternative performance metric, the manual test results were analyzed to find the minimum threshold at which at least one of the sheep was detected in every tested image.

Table 6.11: Manual testing: Detection thresholds at which at least one sheep was detected in every image.

YOLO version:	Minimum threshold:	Avg. false positives:
YOLOv3, split, 5K	32%	0.23
YOLOv3-tiny, 10K	40%	0.17

This shows that using this metric as a measure of success allows higher detection thresholds to be used, which significantly reduces the number of false positives in the results. Furthermore, it is clear from the test results that a small number of images in certain chal-

lenging conditions require a much lower minimum threshold. This means that in less challenging conditions, an even higher threshold can be used, further reducing the number of false positives.

6.5 Combining Thermal and Visual Detection

Thermal and visual detection were both tested separately on corresponding images in the data set. The visual image detection rate was 93.1%, and the thermal image detection rate was 92.0%. The results confirm that the two detection methods have different strengths and weaknesses. For example, visual detection can struggle in low light conditions, when the low ambient temperature makes thermal detection more effective. On the other hand, thermal detection can have problems when the sun is out and the ground temperature is high. This indicates the potential benefits of combining results from both methods.

Table 6.12 shows a comparison of which detection method is most effective under different conditions. In general, it is only worth disregarding the results of one method when it generates too many false positives to be useful. For thermal detection, this is true when the ambient temperature is high, for example daytime in the summer. Visual detection shows more false positives when the sheep are difficult to pick out from the background. This is more noticeable in certain conditions, such as nighttime in the winter, although this effect is not as detrimental as the effects of high ambient temperature on thermal detection. Other than in these types of conditions, combined detection is the most effective method.

Table 6.12: Most effective detection methods under different conditions.

-	**Summer**	**Winter**
Day	Visual	Combined
Night	Combined	Thermal

A simple way of combining thermal and visual detection is to simply count a sheep as found if it is detected by either of the two methods. Using this method, the detection rate increases to 97.9%, which is a 6.4% increase for thermal images and a 5.2% increase for visual images. As discussed in section 7.3, limitations on the available equipment means that the image types cannot be directly combined.

Table 6.13: Performance of combined detection with threshold 10%.

Image type:	Detected sheep:	Avg. false positives:
Thermal	92.0%	0.55
Visual	93.1%	0.86
Combined	97.9%	0.71

This is a significant increase in the detection rate, and it shows that the combination of visual and thermal detection leads to better results for the system overall. It could also

lead to a lower rate of false positives by increasing the detection threshold for both the thermal and visual images, as discussed in subsection 6.1.4.

By taking the alternative metric of detecting at least one sheep in every image as discussed in section 6.4, and applying it to detecting at least one sheep in either the visual or thermal image, the combined system can significantly reduce the amount of false positives in this data set. Table 6.14 shows results from applying this metric to the combined system.

Table 6.14: Manual testing: Detection thresholds at which at least one sheep was detected in every image.

Image type:	Minimum threshold:	Avg. false positives:
Visual	69%	0.09
Thermal	87%	0.10
Combined	97%	0.07

Chapter 7

Further Work

This chapter will cover further work that the authors feel could improve the performance of any future projects in this area, including limitations that the authors encountered during their work on this thesis.

7.1 Expanded Data Set

The data set available consists of relatively few images, and the images are all taken in distinct series. The images in a series all have very similar lighting and terrain conditions, meaning that the variety in the data set is limited. Although the object detection system performs well on the available test data, it is unknown how well it would perform in conditions that differ from conditions in the data set. An expanded data set with more varied conditions would undoubtedly lead to improvements in the performance of the detection system.

An expanded data set would also allow for far more thorough testing and comparison of the performance of the system in different weather, lighting, and terrain conditions. This would allow for a comprehensive comparison and evaluation of how well the system performs under many different conditions, which could lead to further enhancements to the system's overall performance.

7.2 Real-World Testing

Although the system described in this thesis has been tested as thoroughly as possible, there was no opportunity to test it in real-life situations. The thesis was worked on over

two semesters, and there were no sheep available in the field for testing in the first semester. The global COVID-19 pandemic removed any opportunity for testing which was planned in the second semester. This means that the authors were confined to testing on the limited pre-made data set which was available. The authors would have liked to have more opportunities to test the system during development, and this could lead to a better system being developed in the future.

7.3 Better Infrared Camera

Early on in the process of writing this thesis, it was found that the infrared camera on the drone in use had a sensor resolution of only 160x120 px, which limits the maximum altitude and accuracy of detection on the thermal images. More advanced drones have higher-quality thermal cameras, which could lead to a much more effective solution for detecting the sheep, and better integration of thermal and visual detection.

The camera system on the drone which was available for this thesis is not capable of taking both thermal and visual images at the same time. This leads to the images not being similar enough to automatically be used in close combination with each other. In many cases, either the sheep or the drone has moved enough to significantly change the position of objects in the image. This effect is large enough to make an effort to overlay or combine the two types of images very challenging. A drone equipped with two separate camera systems could be capable of more effectively combining the two image types into a single image for the object detection system, and presenting a single set of results to the end-user.

7.4 Fixed-Wing Drone

The drone system which was available for use during the development of this system is a multi-rotor drone with a relatively short flight time. It is therefore only capable of searching a small area before battery replacement is necessary.

Practical use of the system would likely involve a fixed-wing drone, which can achieve much longer flight times and cover a much larger area. The system would need to be tested with such a drone in order to see what impact the constant movement has on the resulting images, and what impact the larger area has on transmitting the images to the control device.

The use of a fixed-wing drone would also require a more stringent license. The new EU regulations coming into force from July 2020 mean that a drone weighing less than 25 kg and flown under 120 meters can be operated in the *"Open"* category, as long as the flight takes place "far from people" [23]. This category only requires an online exam.

Flights over 120 meters or closer to people would fall into the *"Specific"* or *"Certified"* categories, and require much more stringent licenses for the operator and risk assessments for the flights to take place.

7.5 Platform Improvements

The developed platform was mainly used to test the various components of the system and as a proof-of-concept. It can be further improved in multiple ways, such as flight path creation, user feedback, and various bug fixes.

The current flight path creation is very simplistic. The user selects a rectangular area on a map, and the flight path is generated as a grid of waypoints within this area. This could be improved by custom polygon-shaped area selection and user-defined flight paths to allow for more detailed searches. Another improvement would be to integrate flight time into the path selection, and not allowing for search areas larger than what the drone is capable of flying.

User feedback is vital in the development of any application. The platform was not developed with user feedback or ease of use in focus, and as a result of that, there is potential for improvement. This could be improved by displaying the current status of the drone and informing the user of errors that occur.

7.6 Altitude-Based Image Division

The image-splitting system created for this thesis divides the images based purely on their resolution, in order to preserve more of the details in the images. This method was found to successfully improve detection of sheep at higher altitudes. However, it leads to objects in the images having varying sizes, and it also leads to potentially unnecessary detail at lower flight altitudes, where sheep are large enough to be detected merely using the original undivided image.

If the flight altitude of the drone could be recorded along with the image, it could be possible to use altitude as a factor in splitting the images. This would mean that sheep in the images would be close to the same size, and it would avoid unnecessary image division at low altitudes, significantly reducing detection times. Images taken at higher altitudes could be split into smaller parts for detection, whereas images at very low altitudes could be kept in their original resolution.

The drone used for this thesis does not correctly record flight altitude in image data, which leads to this method being challenging to achieve.

Chapter 8

Conclusion

The objective of this project was to investigate an optimal way to create a system for automated detection of sheep. In order to investigate this problem, a system was created and tested.

With regards to Research Question 1 in section 1.1, it was found that the configuration with the best overall accuracy was using YOLOv3 on split images, with an mAP of 98.37%. However, using YOLOv3-tiny on the split images has an mAP only slightly lower, but with significantly higher speed. The manual testing which was performed showed that YOLOv3-tiny has very similar overall detection accuracy compared to YOLOv3, and even performs better on high-altitude images. This is in accordance with the results by Van Gemert et al., who found that accuracy in this type of animal detection can be different from reported overall accuracy rates for object detection systems in the literature [15].

Using the test data available, it was found that the split image detection system finds at least one of the sheep in every image up to a high detection threshold, meaning that false positives can be reduced by using this method as a measure of successful detection. However, real-world performance is still uncertain given the lack of opportunities for testing and developing the system in real conditions, and using a detection threshold which is too high would mean that whole groups of sheep could be missed by the system. More testing would be needed to confirm the usefulness of this method in real-world applications.

The combination of thermal and visual detection methods was found to be very effective in increasing the accuracy of detecting sheep. Limitations with the available equipment meant that more complex methods for combining the data types could not be employed, but a simple combination of the detection results from separate analysis of the images still proved very effective in testing.

With regard to Research Question 2, it was found that the main limiting factor for altitude was the resolution of the thermal camera. The camera which was available for this thesis

has a very low resolution, which severely limits the altitude at which thermal detection can function. However, if thermal detection is disregarded, the drone can be flown up to the legal maximum of 120 meters and still detect the sheep, because of the enhanced resolution provided by the method of splitting the images. Detection at higher altitudes using this method is limited by the resolution of the camera, as the image can be divided into more parts in order to successfully detect smaller objects.

A grid pattern is an optimal way to place the waypoints, because it is a way to achieve complete coverage of the area. The drone used for this thesis is a multi-rotor drone which can stop at each waypoint. This ensures that the images are free of motion blur, and is therefore an optimal pattern for this type of drone. If a fixed-wing drone were to be used, more care would have to be taken in order to create a pattern that is optimized for this type of flight, where the drone has limitations on turning circle and minimum flight speed.

For Research Question 3, extensive testing was performed on different hardware configurations, as shown in section 6.2. In general, there is a trade-off between hardware cost, accuracy, and detection time. YOLOv3-tiny can be fast enough to run the system on a CPU, with run times per image between 6-8 seconds on a desktop CPU and about 18 seconds on a laptop. Depending on flight altitude, this can allow the results to be analyzed faster than the flight time between each waypoint.

Running the system locally allows for detection to be performed without an internet connection. This removes geographic restrictions for areas with no cellular network reception. In general, running the system locally on a laptop is significantly slower than running on a remote server. However, certain configurations can still be fast enough to allow for practical use on such a system.

In conclusion, with regards to the main Research Question this thesis found that an optimal way to detect sheep in the field is to combine visual and thermal object detection using the methods discussed above. If an internet connection is available the object detection can be run remotely on a server, which is the fastest overall approach of the tested configurations. However, YOLOv3-tiny is also fast enough to be used on a laptop computer in the field.

References

[1] B. Pietsch, *Global drone market estimated to reach $14 billion over next decade: Study*, 2019. [Online]. Available: https://www.reuters.com/article/us-usa-security-drones/global-drone-market-estimated-to-reach-14-billion-over-next-decade-study-idUSKCN1UC2MU (visited on 02/06/2020).

[2] P. Tripicchio, M. Satler, G. Dabisias, E. Ruffaldi, and C. A. Avizzano, "Towards Smart Farming and Sustainable Agriculture with Drones," *2015 International Conference on Intelligent Environments*, 2015.

[3] J. H. Muribø, "Locating Sheep with YOLOv3," Master's thesis, Norwegian University of Science and Technology, 2019.

[4] *LabelBox*. [Online]. Available: https://labelbox.com (visited on 09/26/2019).

[5] L. Asheim and 1. Mysterud, "The norwegian sheep farming production system," *Options Méditerranéennes: Série A. Séminaires Méditerranéens*, 1999.

[6] G. Austrheim, L.-J. Asheim, G. Bjarnason, J. Feilberg, A. M. Fosaa, Ø. Holand, K. Høegh, I. S. Jónsdóttir, B. Magnússon, L. E. Mortensen, A. Mysterud, E. Olsen, A. Skonhoft, G. Steinheim, and A. G. Thórhallsdóttir, "Sheep grazing in the north-atlantic region - a long term perspective on management, resource economy and ecology," *NTNU Vitenskapsmuseet Rapport Zoologisk Serie*, 2008.

[7] A. M. Sibbald and R. J.Hooper, "Sociability and the willingness of individual sheep to move away from their companions in order to graze," *Applied Animal Behaviour Science*, 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0168159103002983.

[8] F. Giones and A. Brem, "From toys to tools: The co-evolution of technological and entrepreneurial developments in the drone industry," *Business Horizons*, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0007681317301210.

[9] *Civil Aviation Authority Norway - Droneguide*, 2019. [Online]. Available: https://luftfartstilsynet.no/globalassets/dokumenter/dronedokumenter/droneplakat_engelsk_print_v3_2019.pdf (visited on 02/13/2020).

[10] Z. Zou, Z. Shi, Y. Guo, and J. Ye, "Object Detection in 20 Years: A Survey," *IEEE TPAMI*, 2019.

[11] R. Geirhos, C. R. M. Temme, J. Rauber, H. H. Schütt, M. Bethge, and F. A. Wichmann, "Generalisation in humans and deep neural networks," in *Advances in Neural Information Processing Systems 31 (NIPS 2018)*, 2018.

[12] L. Svendsen, "Gjenfinning av Sau på Utmarksbeite," Master's thesis, Norwegian University of Science and Technology, 2019.

[13] C. Burke, M. Rashman, S. Wich, A. Symons, C. Theron, and S. Longmore, "Optimizing observing strategies for monitoring animals using drone-mounted thermal infrared cameras," *International Journal of Remote Sensing*, 2018. DOI: 10.1080/01431161.2018.1558372. [Online]. Available: http://dx.doi.org/10.1080/01431161.2018.1558372.

[14] J. Cukor, J. Bartoška, J. Rohla, J. Sova, and A. Machálek, *Use of aerial thermography to reduce mortality of roe deer fawns before harvest*, https://doi.org/10.7717/peerj.6923, 2019.

[15] J. C. van Gemert, C. R. Verschoor, P. Mettes, K. Epema, L. P. Koh, and S. Wich, "Nature Conservation Drones for Automatic Localization and Counting of Animals," in *Computer Vision – ECCV 2014 Workshops*, 2014.

[16] P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 2008.

[17] F. S. Khan, R. M. Anwer, J. van de Weijer, A. D. Bagdanov, M. Vanrell, and A. M. Lopez, "Color attributes for object detection," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012.

[18] T. Malisiewicz, A. Gupta, and A. A. Efros, "Ensemble of exemplar-svms for object detection and beyond," in *2011 International Conference on Computer Vision*, 2011.

[19] J. Redmon, D. Santosh, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," *arXiv*, 2016.

[20] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," *arXiv*, 2016.

[21] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv*, 2018.

[22] D. Floreano and R. J. Wood, *Science, technology and the future of small autonomous drones*, 2015. [Online]. Available: https://www.nature.com/articles/nature14542 (visited on 02/13/2020).

[23] *CAP 1789 - The EU UAS Regulation Package - Outline*, 2020. [Online]. Available: https://publicapps.caa.co.uk/docs/33/CAP1789%20April%202020.pdf (visited on 05/28/2020).

[24] *DJI Mavic 2 Enterprise Dual Specs*. [Online]. Available: https://www.dji.com/no/mavic-2-enterprise/specs (visited on 02/03/2020).

[25] J. Zou, Y. Han, and S.-S. So, "Overview of Artificial Neural Networks," in *Artificial Neural Networks - Methods and Applications*, Humana Press, 2008.

[26] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014.

[27] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, "Convolutional neural networks: An overview and application in radiology," *Insights into Imaging*, 2018. [Online]. Available: https://link.springer.com/article/10.1007/s13244-018-0639-9.

[28] A. Vidhya, *YOLO v3 theory explained*, 2019. [Online]. Available: https://medium.com/analytics-vidhya/yolo-v3-theory-explained-33100f6d193 (visited on 03/24/2020).

[29] *YOLO: Real-Time Object Detection*. [Online]. Available: https://pjreddie.com/darknet/yolo/ (visited on 02/17/2020).

[30] A. Rosebrock, *YOLO and Tiny-YOLO object detection on the Raspberry Pi and Movidius NCS*. [Online]. Available: https://www.pyimagesearch.com/2020/01/27/yolo-and-tiny-yolo-object-detection-on-the-raspberry-pi-and-movidius-ncs/ (visited on 03/20/2020).

[31] *ImageNet*. [Online]. Available: http://www.image-net.org/ (visited on 02/17/2020).

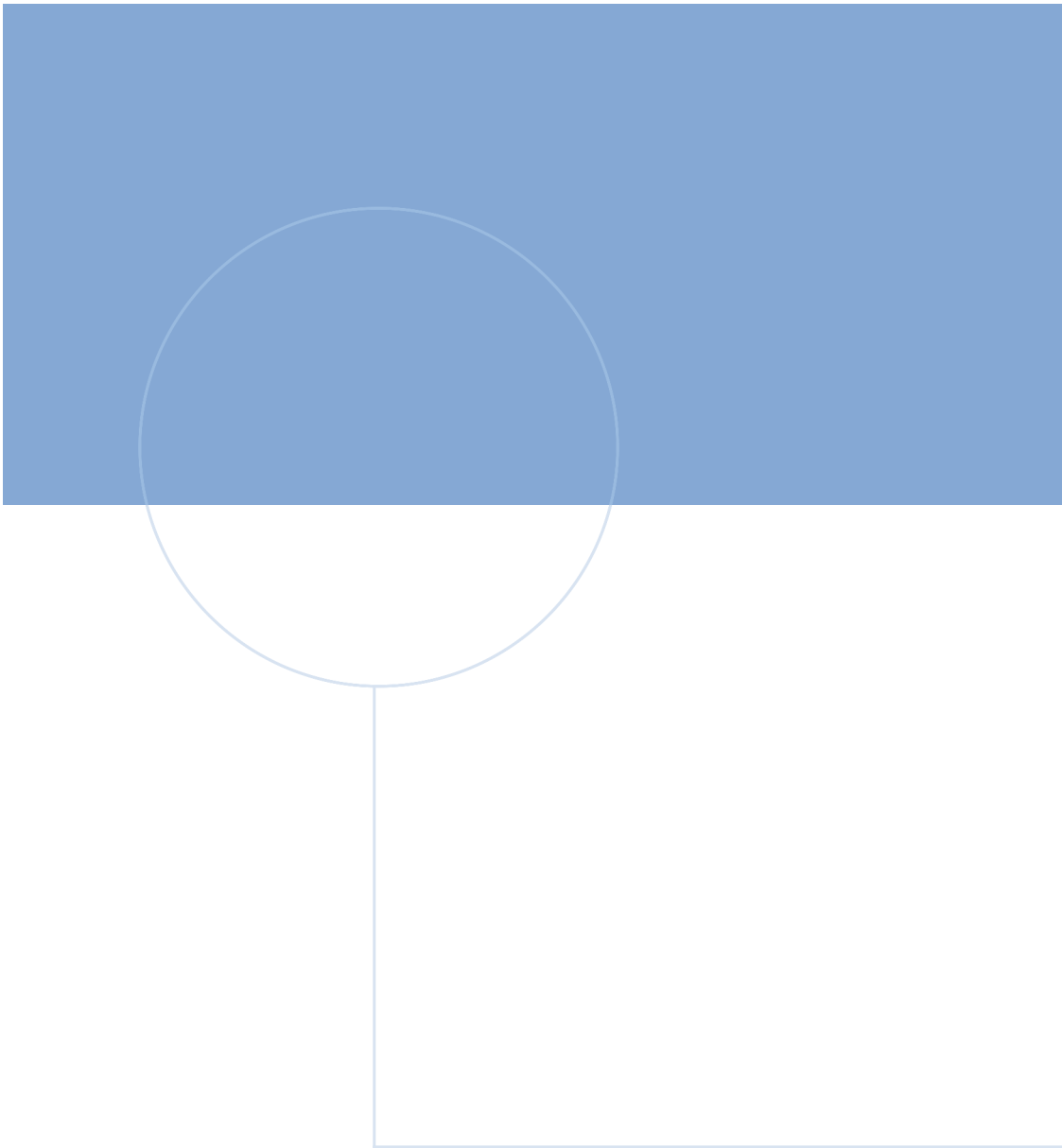
[32] A. Kathuria, *What's new in YOLO v3?* 2018. [Online]. Available: https://towardsdatascience.com/yolo-v3-object-detection-53fb7d3bfe6b (visited on 02/17/2020).

[33] A. Rosebrock, *Intersection over Union (IoU) for object detection*, 2016. [Online]. Available: https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (visited on 02/13/2020).

[34] *Client/Server Architecture*, 2018. [Online]. Available: https://www.techopedia.com/definition/438/clientserver-architecture (visited on 02/13/2020).

[35] *IDUN Hardware*. [Online]. Available: https://www.hpc.ntnu.no/idun/hardware (visited on 03/03/2020).

[36] AlexeyAB, *Darknet fork: When should I stop training*. [Online]. Available: https://github.com/AlexeyAB/darknet#when-should-i-stop-training (visited on 03/05/2020).



NTNU

Kunnskap for en bedre verden